This is the authors' original unrevised version of the manuscript. The final version of this work (the version of record) is published by Springer in the book *Guide to Brain-Computer Music Interfacing*, ISBN 9781447165835. This text is made available on-line in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

CHAPTER 12

Creative Music Neurotechnology with *Symphony of Minds Listening*

Eduardo Reck Miranda[1], Dan Lloyd[2], Zoran Josipovic[3] and Duncan Williams[1]

Abstract
A better understanding of the musical brain, combined with technical advances in Biomedical Engineering and Music technology are pivotal for the development of increasingly more sophisticated Brain-Computer Music Interfacing (BCMI) systems. BCMI research has been very much motivated by its potential benefits to the health and medical sectors, as well as to the entertainment industry. However, we advocate that the potential impact on musical creativity of better scientific understanding of the brain, and the development of increasingly sophisticated technology to scan its activity, should not be ignored. In this chapter we introduce an unprecedented new approach to musical composition, which combines brain imaging technology, musical Artificial Intelligence and Neurophilosophy. We discuss *Symphony of Minds Listening*, a composition for orchestra in three movements, based on the fMRI scans taken from three different people while they listened to the second movement of Beethoven's *Seventh Symphony*.

[1] Interdisciplinary Centre for Computer Music Research (ICCMR)
Plymouth University
Plymouth, PL4 8AA
United Kingdom
e-mail: {eduardo.miranda, ducan.williams}@plymouth.ac.uk

[2] Department of Philosophy, Program in Neuroscience
Trinity College
Hartford, CT 06106
USA
e-mail: Dan.Lloyd@Trincoll.edu

[3] Psychology Department
New York University
6 Washington Place, Room 482a
New York, NY 10003
USA
e-mail: zoran@nyu.edu

12.1 Introduction

BCMI research has been very much motivated by its potential benefits to the health and medical sectors, as well as to the entertainment industry. Yet, advances in the field tend be assessed in terms of medium, rather than content. For instance, let us consider the field of Music Technology. Much has been said on the improvement of technology for music recording and distribution, from vinyl records and K7 tapes to CDs and the Internet. However, not much is said on the impact of these media to creative processes. Have these media influenced the way in which music is composed? Likewise, not much has been said on the creative potential of BCMI technology. Might it lead to new ways to make music, or to the emergence of new kinds of music?

We believe that the potential impact on musical creativity of better scientific understanding of the brain, and the development of increasingly sophisticated technology to scan its activity can be huge. Musicians have an unprecedented opportunity today to develop new approaches to composition that would have been unthinkable a few years ago.

In this chapter we introduce an unprecedented new approach to musical composition, which combines brain imaging technology (Bremmer 2005), musical Artificial Intelligence (AI) (Miranda 2000), and new philosophical thinking emerging from Neurophilosophy (Churchland 2007). The first outcome of this approach is *Symphony of Minds Listening*, an experimental composition for orchestra in three movements, based on the fMRI scans taken from three different people while they listened to the second movement of Beethoven's *Seventh Symphony*: a ballerina, a philosopher (co-author Dan Lloyd) and a composer (co-author Eduardo R. Miranda). In simple terms, we deconstructed the Beethoven movement to its essential elements and stored them with information representing their structural features. Then, we reassembled these elements into a new composition with a twist: the fMRI information influenced the process of reassembling the music.

The chapter begins with a discussion on the philosophical ideas behind the work. Next, before delving into more technical details, it gives an overview of the compositional approach we have been developing. It follows with an introduction to the brain scanning and data analysis methods. Then, it introduces MusEng, the system that we developed to deconstruct and re-compose music and demonstrate the core processes behind the composition of *Symphony of Minds Listening*.

12.2 Neurophilosophy of Music

The human brain is allegedly the most complex object known to mankind: it has circa one hundred billion neurones forming a network of an estimated one quadrillion connections between them. The amount of information that circulates through this network is huge. The operation of individual neurones is fairly well understood nowadays, but an important piece of the jigsaw is missing: the way they cooperate in ensembles of millions has been fiendishly difficult to understand. This piece of the puzzle is important because it most probably holds the key to unlock our understanding the origins of the mind.

There has been a school of thought, which considered that the mind is divorced from the brain. What is more, it has been suggested that minds would not even need brains to exist. Although the separation between mind and brain still has currency in some circles, nowadays it is common sense to consider the mind as resulting from the functioning of the brain. However, we do not have a clear understanding of how brain activity actually gives rise to the mind.

Much research is being developed from a number of approaches all over the globe to understand how the brain gives rise to the mind. Our research is looking into establishing a musical approach to understand the brain. We believe that the brain can be viewed as a colossal, extraordinarily large symphonic orchestra and the mind as a correspondingly complicated symphony. The 'mind as music' hypothesis is explored in length in (Lloyd 2011).

At Plymouth University's Interdisciplinary Centre for Computer Music Research we are looking into the relationship between music and a specific aspect of our mind: *emotions*. We hope to be able to determine which aspects of a musical composition elicit specific emotions on listeners. The hypothesis is that if one can predict which musical features are likely to cause the feeling of, say, joy or sadness, then it might be possible to build technology that would allow new music to steer our emotions more effectively. For example, it would be highly beneficial for humankind if physicians could have the option to prescribe a musical composition as part of the treatment to help take a patient out of depression. Not unlike chemists, future musicians could be trained with the skill to compose with specific musical ingredients aimed at inducing particular affect in listeners. Our work is aimed at making this ambitious dream a reality, but the challenges to achieve this are not trivial.

Similar to the fact that we have unique fingerprints, which differ from person to person, our brains are also unique. Indeed, the mechanisms whereby we make sense of music differ from person to person. Even though all human brains share a common basic plan, the detailed neurological circuitry differs from one person to another. Unlike our fingerprints, however, our brain circuits are continually changing and this makes scientific research into unveiling how the brain functions rather challenging. Paradoxically, it seems that the more

we study the brain, the more difficult it becomes to draw firm conclusions. A balance needs to be established between identifying the commonalities and acknowledging the differences of our brains. *Symphony of Minds Listening* is inspired by the later: it is an artistic expression of how different brains construct their own unique reality.

12.3 An Overview of the Approach

Functional magnetic resonance imaging (fMRI) is a procedure that measures brain activity by detecting associated changes in blood flow. The measurements can be presented graphically by colour-coding the strength of activation across the brain. Figure 12.1 shows a typical representation of an fMRI scan of a person listening to music, displaying the activity of the brain at a specific window of time. In this case, each time window lasts for two seconds. The figure shows 14 planar surfaces, or slices, from the top to the bottom of the brain, and the respective activity detected in these areas. Figure 12.2 is an example of an artistic 3D rendition of such an fMRI scan. It shows different areas of the brain, represented by different colours (that is, shades of grey), responding in a coordinate manner to the music.

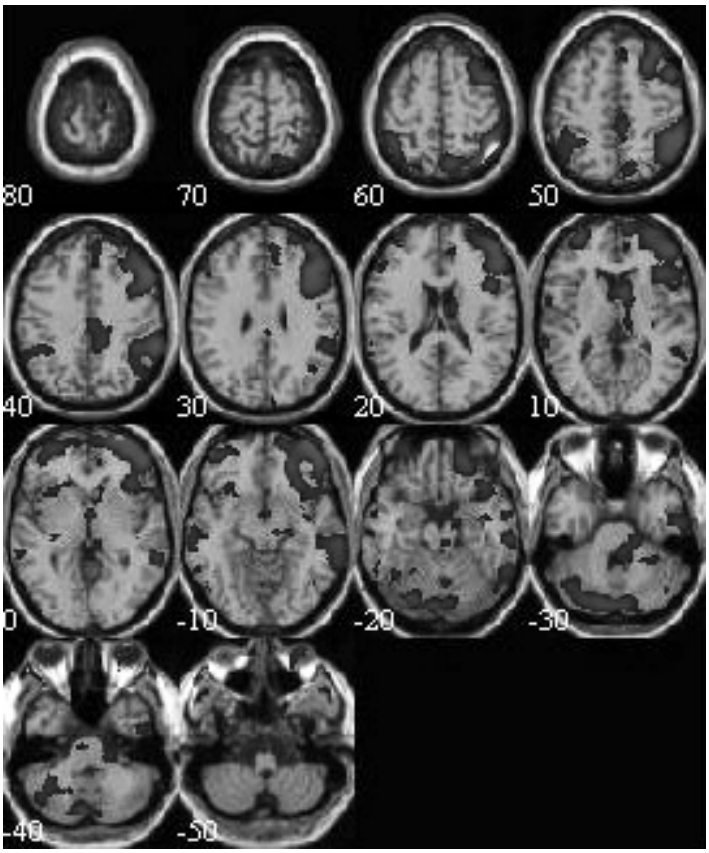

Figure12. 1: A typical representation of an fMRI scan, showing 14 slices of the brain. The actual scanning for this project comprised 36 slices snapshots taken every 2 seconds.

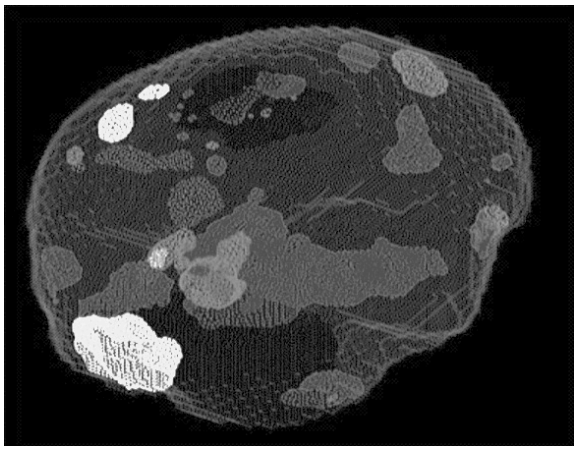

Figure 12.2: An artistic 3D rendering of an fMRI scan.

Each scanning session generated sets of fMRI data, each of which associated to a measure of the second movement of Beethoven's 7th symphony. This is shown schematically in Figure 12.3.

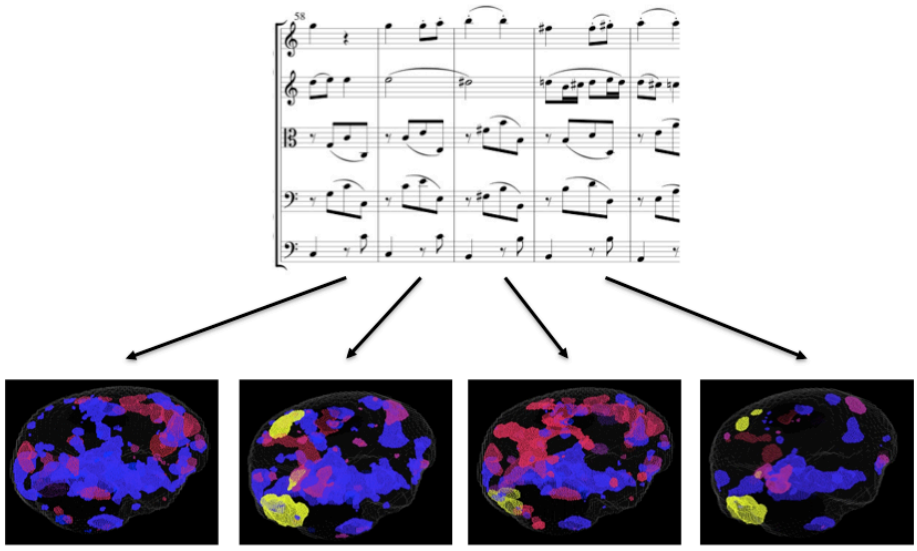

Figure 12.3: The result of a scanning section is a set of fMRI data for each measure of Beethoven's piece.

Firstly, the movement was deconstructed by means of MusEng, a piece of software, which extracted information about the structure of the Beethoven

piece. Then, we programmed MusEng to use this information and the fMRI data to generate new musical passages.

During the compositional phase, the fMRI information was used on a measure-by-measure basis to influence the composition. This procedure, which is shown schematically in Figure 12.4, involved diverse modes of data processing and transformation of Beethoven's music. The resulting musical passages bore varied degrees of resemblance to the original.

Not surprisingly, the fMRI scans differed amongst the three listeners. Therefore, brain activity from three different minds yielded three different movements in the resulting composition that resemble the original in varied ways. The instrumentation is the same as for Beethoven's original instrumentation and each movement is named after the respective person that was scanned:

- 1st Movement: *Ballerina*
- 2nd Movement: *Philosopher*
- 3rd Movement: *Composer*

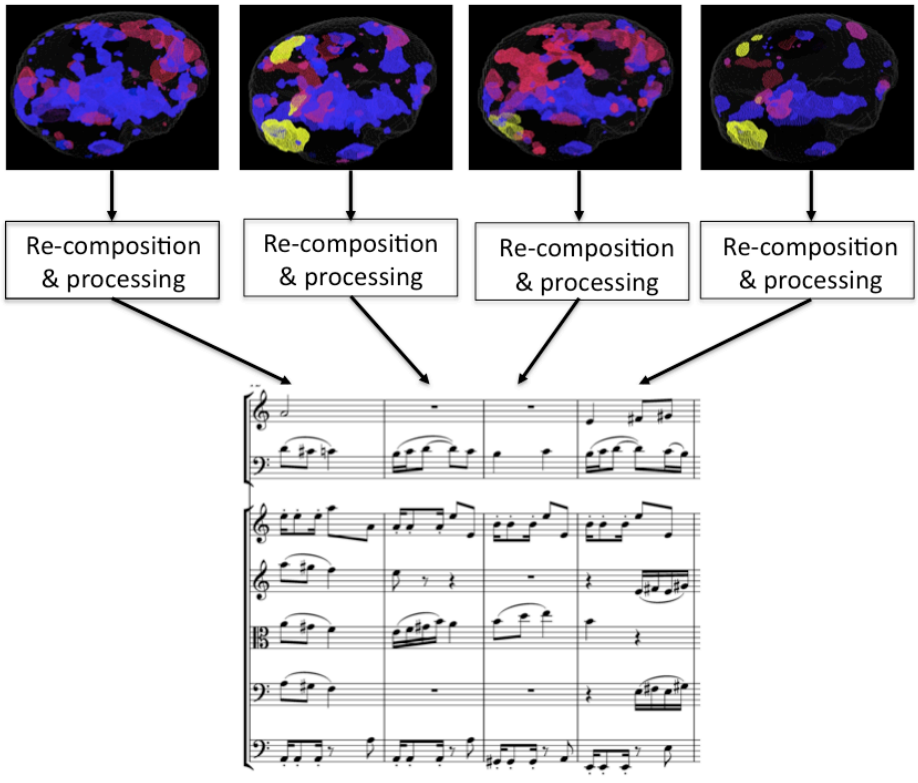

Figure 12.4: The fMRI data inform the re-assemblage of the piece.

12.4 Brain Scanning: Materials and Methods

The brain images were collected using equipment and parameters that are typical in Cognitive Neuroscience. The scanner was a Siemens Allegra 3T head-only scanner with a head coil. ("T" stands for Tesla, a measure of magnetic field strength. Contemporary scanners range from 1.5T to 7T.) A T2-sensitive echo planar imaging (EPI) pulse sequence was used to obtain blood oxygenation level-dependent (BOLD) contrasts: TR = 2000 ms, TE = 30 ms, 36 axial slices, 3 x 3 x 3 mm, 64 x 64 matrix in a 192 x 192 mm FOV. That is, each full-brain image took two seconds to collect, to yield 36 image slices of the brain. Each slice comprised 64 x 64 picture elements, known as voxels or volume pixels. Thus, each image comprised approximated 150,000 continuously varying voxels.

Subjects heard the second movement of Beethoven's Seventh Symphony twice. The subjects were instructed to attend to the music with their eyes closed. The fMRI recording began with 30 seconds without music, then 460 seconds of Beethoven, then 18 seconds without music, and finally the same 460 seconds of Beethoven previously heard. Thus each run generated 484 images.

12.5 fMRI Analysis

The raw fMRI scans were first pre-processed following usual procedures for functional neuroimaging. These included correcting for head motion, morphing the individual brains to conform to a standard anatomical atlas, and spatial smoothing, which is a procedure that reduces random fluctuations by calculating a moving average of each voxel in the context of its spatial neighbours. These pre-processing steps were implemented using Statistical Parametric Mapping software (Ashburner et al. 2013).

Each of the 484 images produced 150,000 voxels, which are very complex for direct analysis. Instead, the image series were further processed with Independent Component Analysis, abbreviated as ICA (Stone 2004). Informally, ICA separates ensembles of voxels that oscillate in unison. These are unified as supervoxels representing temporally coherent networks of brain activity. The coloured patches in Figure 12.2 are examples of independent components. A total of 25 components were calculated for the three subjects in the experiment.

In order to select which of these components might be musically significant, the activity of each component during the first pass through the Beethoven listening was compared to that same component during the second pass. If these two segments of a component time series were correlated, we hypothesized that the activity was at least partly musically driven, since the stimulus, that is, the music, would be identical at the corresponding time points in the two passes through the music. Although 25 independent component time series were identified, only the strongest 15 were selected to influence the compositional process. The order of strength of the selected 15 ICA components is as follows: 25, 15, 14, 8, 5, 10, 11, 18, 6, 2, 4, 1, 17, 16 and 13. The time series were normalized to range from 0 to 9. As a last step,

the varying components were resampled to match the timing of the Beethoven score measure by measure. Thus, each time point was indexed to a measure of the Beethoven score. The movement comprises 278 measures, therefore each ICA component comprises a time series of 278 values, ranging from 0 (meaning lowest fMRI intensity) to 9 (highest fMRI intensity). As an example, Table 12.1 shows the values of the first 5 strongest ICA components (that is, 25, 15, 14, 8 and 5) for the first 10 measures of Beethoven's music, yielded by the fMRI of the subject 'composer' the during the first listening pass in the scanner.

Beethoven Measure	ICA 25	ICA 15	ICA 14	ICA 8	ICA 5
1	7	5	5	5	2
2	5	5	8	5	8
3	7	3	5	5	6
4	5	8	3	5	2
5	5	7	4	4	4
6	6	6	4	5	3
7	7	8	5	6	3
8	4	6	3	4	3
9	6	6	4	5	4
10	5	7	5	5	3

Table 12.1: The values of the strongest 5 ICA components for the first 10 measures of Beethoven's music yielded by the subject 'composer'.

To accompany the premiere of *Symphony of Minds Listening*, the ICA data were animated on a large screen projection behind the orchestra. The whole brain appeared as a transparent frame, derived from a standard anatomical template. Within this image, each component was assigned a distinct color, and brightened and faded according to the intensity of component activity at each time point. The animations were created using Matlab software (MathWorks 2013), using custom-made functions. The remaining of this chapter focuses on the compositional process and the MusEng software.

12.6 The Compositional Process

The actual composition of *Symphony of Minds Listening* is primarily the work of the first co-author and involved a number of creative stages and practices, some which were not systematically documented. That is to say, the compositional process involved manual and computer-automated procedures.

There generally are two approaches to using computer-generated materials in composition, which we refer to as the *purist* and *utilitarian* approaches, respectively. The purist approach to computer-generated music tends to be more concerned with the correct application of the rules programmed in the system, than with the musical results per se. In this case, the output of the

computer tends to be considered as the final composition. The composer would not normally modify the music here, as this would meddle with the integrity of the model or system. At the other end of the spectrum is the utilitarian approach, adopted by those composers who consider the output from the computer as raw materials for further work. Here composers would normally tweak the results to fit their aesthetic preferences, to the extent that the system's output might not even be identifiable in the final composition. Obviously, there is a blurred line dividing these two approaches, as practices combining aspects of both are commonly found. *Symphony of Minds Listening* tends towards the utilitarian approach.

The composition of the piece evolved in tandem with the development of MusEng. MusEng was programmed with Artificial Intelligence to learn musical information from given examples and use this information to generate new music. Incidentally, a few of MusEng's functionalities were firstly applied manually to compose a section of the piece, before they were implemented in software to aid the composition of other sections. Indeed, a number of compositional processes did not make it into the software on time. The symphony had a deadline to be delivered for its premiere in February 2013, at Peninsula Arts Contemporary Music Festival, in Plymouth, UK. The software development, however, is still in progress. And other pieces are planned and the compositional approach is being refined as we write this chapter.

For a discussion on how science and technology can inform and inspire the act of musical composition the reader is referred to (Miranda 2013) and (Miranda 2014). Both references advocate the use of computers as assistants to the creative process, rather than as autonomous composing machines.

For the composition of *Symphony of Minds Listening*, the first step was to deconstruct the score of Beethoven's piece into a set of basic materials for processing. These materials were subsequently given to MusEng as input for a machine learning phase, which will be explained in more detail in the next section of this chapter.

First of all, Beethoven's piece was divided into 13 sections:

- Section 1: from measure 1 to measure 26
- Section 2: from measure 26 to measure 50
- Section 3: from measure 51 to measure 74
- Section 4: from measure 75 to measure 100
- Section 5: from measure 101 to measure 116
- Section 6: from measure 117 to measure 138
- Section 7: from measure 139 to measure 148
- Section 8: from measure 149 to measure 183
- Section 9: from measure 184 to measure 212
- Section 10: from measure 213 to measure 224
- Section 11: from measure 225 to measure 247
- Section 12: from measure 248 to measure 253
- Section 13: from measure 254 to measure 278

The 13 sections informed the overarching form of the 3 movements of the new symphony. This provided a template for the new piece, which preserved the overall form of the original Beethoven movement. Indeed, MusEng did not learn the whole Beethoven piece at once. Rather, it was trained on a section-by-section basis and the musical sequences for the respective new sections of the new movements where generated independently from each other. For instance, Section 1 of the movement *Ballerina* has 26 measures and was composed based on materials from the first 26 measures of Beethoven's music. Next, Section 2 has 24 measures and was composed based on materials from the next 24 measures (26 - 50) of Beethoven's music, and so on.

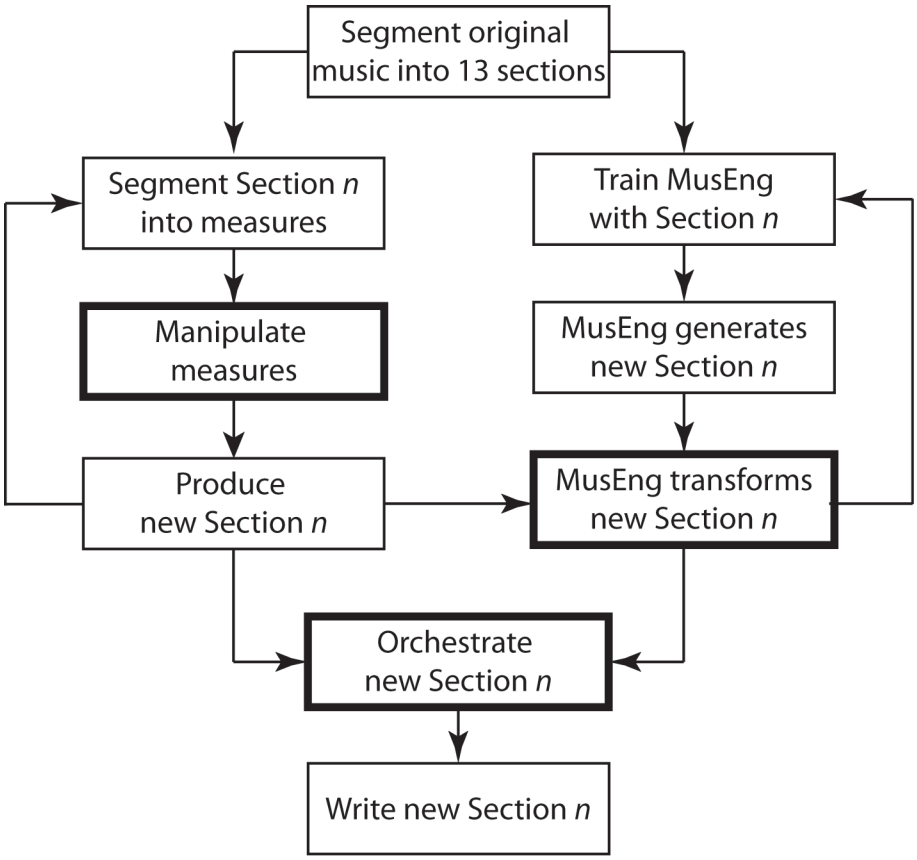

Figure 12.5: Block diagram of the overall compositional process.

A block diagram portraying the compositional procedures is shown in Figure 12.5. The blocks with thicker lines represent procedures that can be influenced and/or controlled by the fMRI. After the segmentation of the music into 13 sections the flow of action bifurcates into two possibilities: manual handling of the segments (left hand side of Figure 12.5) and computerised handling with MusEng (right hand side of Figure 12.5). A discussion of manual handling is beyond the scope of this chapter, but as an example we can show the transformation of Section 1 of Beethoven's original into the opening section of *Ballerina*. Figure 12.6 shows the first 10 measures of Beethoven's music focusing on the parts of the violas, violoncellos and double basses. Figure 12.7 shows how those measures were recomposed to form 10

measures for the opening of the first movement of the new symphony. Note the visible rhythmic transformation of measures 4, 6, 8 and 10.

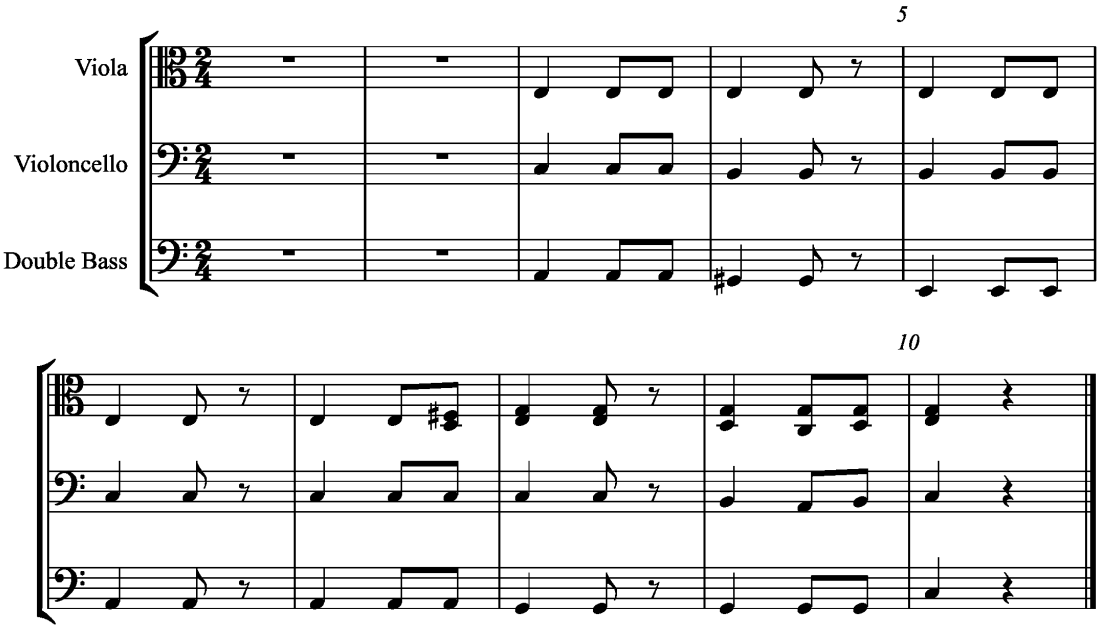

Figure 12.6: The first 10 measures of Section 1 of Beethoven's music, showing the viola, violoncello and double bass parts.

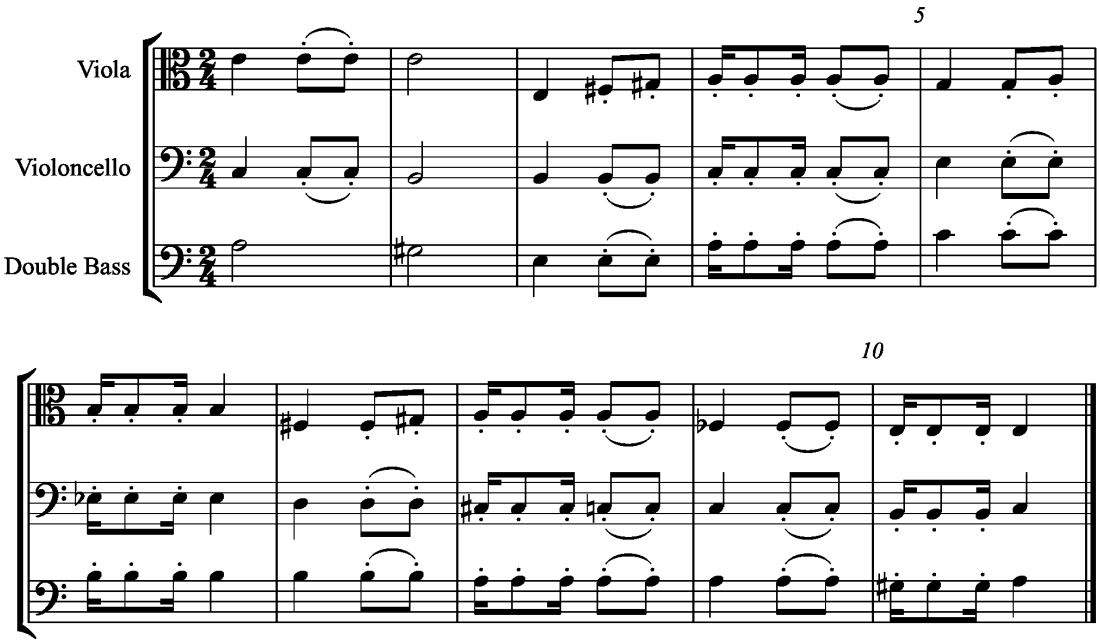

Figure 12.7: Ten measures from the opening of *Ballerina*, the first movement of *Symphony of Minds Listening*, showing the viola, violoncello and double bass parts.

The path on the right hand side of the block diagram in Figure 12.5 represents the computer handling of the segments with MusEng. This will be explained in more detail in the next section.

Finally, once a new segment has been generated, it is orchestrated and appended to the respective score of the new movement accordingly. The fMRI occasionally influenced the instrumentation and the orchestration. For instance in *Philosopher*, the second movement, different ICA components were associated to groups of instruments of the orchestra (e.g., ICA 25 = violins and violas, ICA 15 = trumpets and horns, ICA 14 = oboes and bassoons, and so on); these associations changed from section to section. Then, for example, if the flute is to play in a certain measure *x* of *Philosopher*, the ICA activation value of the respective component in measure *x* of Beethoven's music would define how the flute player should produce the notes. We defined various tables mapping ICA activations to instrumental playing techniques and other musical parameters. For instance, Table 14.2 shows a mapping of ICA activations onto musical dynamics.

Activation	Dynamics
0	*ppp*
1	*ppp*
2	*pp*
3	*p*
4	*mp*
5	*mf*
6	*f*
7	*ff*
8	*fff*
9	*fff*

Table 12.2: Mapping ICA activation values onto musical dynamics.

As a hypothetical example, let us consider a case where the flutes would play the sequence shown in Figure 12.8 in measures 5, 6 and 7 of the third movement*: Composer*. If we assume that the flute is associated to ICA 8, then according to the values shown in Table 12.1, the activations for measure 5, 6 and 7 are equal to 4, 5 and 6, respectively. Thus, the dynamics attributed to these 3 measures would be as shown in Figure 12.9.

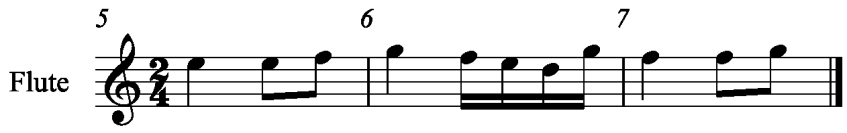

Figure 12.8: Three new measures for the flutes.

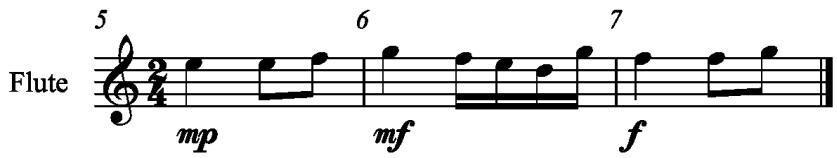

Figure 12.9: The measures from Figure 12.8 with added dynamics informed by fMRI information.

12.7 The Musical Engine: MusEng

MusEng has three distinct phases of operation: a *learning phase*, a *generative phase* and a *transformative phase*.

The learning phase takes a musical score and analyses it in order to determine a number of musical features. A dataset comprising these features and rules representing the likelihood of given features appearing in the data are then stored in memory. At the generative phase, these data inform the generation of new sequences, which should ideally resemble the sequences that were used to train the system in the first phase. Finally, at the transformative phase, the outcome from the generative phase is modified according to a number of transformation algorithms. It is in this final phase that the fMRI information is used to influence the resulting music. Note that we are not interested in a system of rules that reproduces an exact copy of the original music. Rather, we are interested in producing new music that resembles the original. Hence the transformative phase was added to further modify the results from the generative phase. The role of fMRI information is to control the extent of the transformations. Essentially, stronger activity in a given ICA component of the fMRI data results in larger amounts of transformation in the musical outcome.

MusEng reads and outputs musical scores coded in the MIDI format. MIDI (Musical Instrument Digital Interface) is a protocol developed in the 1980's, which allows electronic instruments and other digital musical tools to communicate with one another. Music notation software normally has an option to save and read files in this format. This is useful because it is straightforward to make a MIDI file representing the Beethoven symphony to train the system. MusEng outputs can be loaded into any suitable music notation software for further work and adjustments.

MusEng only processes monophonic musical sequences, that is, sequences of one note at a time. Obviously, Beethoven's movement is a polyphonic orchestral symphony. To circumvent MusEng's monophonic limitation we developed two approaches to process the music. In the first approach we train the system with the part of one instrumental voice of the orchestra at a time (violins, violoncellos, etc.) and then we generate sequences for those respective parts individually. In the second approach we reduce the orchestral music to one monophonic voice and then generate various monophonic sequences, which are subsequently orchestrated.

12.7.1 Learning Phase

MusEng implements an adapted version of iMe (short for Interactive Musical Environments), a system developed at Plymouth University's Interdisciplinary Centre for Computer Music Research with Marcelo Gimenes (Miranda and Gimenes 2011). MusEng takes a MIDI file as an input and extracts the following 5 features from the encoded music:

- Pitches of the notes
- Melody directions between successive notes in a sequence
- Melody intervals; i.e., the amount of change between the pitches of successive notes in a sequence
- Note durations
- Modalities implied by groups of notes in a sequence

These features are stored as event-based vectors, referred to as *musicodes*. Table 12.3 shows the musicodes for the first two measures of the musical excerpt shown in Figure 12.10.

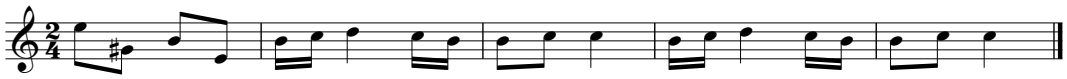

Figure 12.10: An example of a musical sequence.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Melody direction	0	-1	+1	-1	0	+1	+1	-1	-1
Melody interval	0	5	2	4	0	1	1	1	1
Event duration	120	120	120	120	60	60	120	60	60
Note pitch	E5	G#4	B5	E4	B5	C5	D5	C5	B5
Modality	E Maj A harmonic min				A min C Maj				

Table 12.3: Musicodes for the first two measures of the musical sequence in Figure 12.10. The rows correspond to the event number, or in this case, number of notes in the sequence: the first two measures comprise a total of 9 notes.

Melody direction can be -1, 0, or +1, referring to descending, motionless, or ascending movement. The current note in a sequence is compared with the previous note; the very first note in a sequence therefore returns a value equal to 0. Melody intervals are represented in terms of half steps, which are also calculated with reference to the current note's distance from the previous

note. Again, the first note in the sequence returns a value equal to 0. With note durations, the value 240 is assigned to quarter notes, and other durations are calculated with reference to this value; e.g., half notes are equal to 480 and eighth notes are equal to 120. Values for pitches are readily extracted directly from the corresponding MIDI code; for instance MIDI 21 = Note A0, MIDI 23 = Note B0, MIDI 24 = Note C1, and so on.

In general, the number -2 is used to represent the absence of data in a musicode vector. Thus, the note pitch musicode for a musical rest would be equal to -2. With respect to the implied modality of segments, the system creates a label specifying a tonal pattern and indicates when the estimation is ambiguous. For example, in the first measure of the music shown in Figure 12.10, the system sees E, G#, and B, as an E Major chord, but the G# has also implied A harmonic minor.

As we shall see below, MusEng builds a musical memory in terms of small segments of music. Ideally, the system would segment the music based on perceptual criteria. The original iMe system sported such a method, inspired by Gestalt psychology (Eysenck and Keane 2005). However, for this project we programmed MusEng to segment the music according to a user specified number of measures; e.g., every measures, or every two measures, or every three and so on. The rationale for this decision is that we wanted to synchronise the fMRI analysis to the input score by handling the fMRI data on a measure-by-measure basis, as it was shown schematically in Figures 12.3 and 12.4. Therefore, it made more sense to establish the measure as a reference value to segment the music.

MusEng's memory consists of a series of Feature Tables (FTs), which comprise vectors of musicodes for material that the system has been exposed to. As the musicodes are extracted from incoming measures, the system may or may not create new FTs, depending on whether the respective musicodes have already been seen by the system. If a certain vector of musicodes is identical to one that has been previously seen by the system, then the system updates the relevant FT by increasing a weighting factor, represented by the variable ω (Equation 12.1). This variable is generated by summing the total number of FTs, and then dividing the number of instances of each individual FT by the total. In essence this becomes a simple moving average. In Equation 12.1 the value of ω indicates the weighting factor associated with a given FT (or FT). The variable x represents the number of instances of a given FT in the series, and n the total number of FT in the series.

$$\omega_{(x)} = \frac{\Sigma \mathrm{FT}_{(x)}}{\Sigma \mathrm{FT}_{(n)}} \tag{12.1}$$

This moving average has the effect of lowering the value of ω for vectors of musicodes that do not appear as often as more frequently ones, in the same way that it raises the value of ω for more commonly used vectors, to a

maximum value of 1.0. The value of ω informs the probability of a given musical segment being generated later on by the system. Typically, a decrease in the value of ω causes the system to 'forget' to utilise the corresponding FT entry in the subsequent generative phase.

In order to illustrate how MusEng's memory is built, let us examine a hypothetical run through the sequence previously shown in Figure 12.10, commencing with an empty memory. The first measure (Figure 12.11) is analysed and the respective musicodes are generated. For the sake of clarity, this example will focus on three of the five features: melody direction (*dir*), melody interval (*int*), and event duration (*dur*).

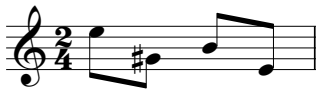

Figure 12.11: The first measure for the example analysis.

MusEng creates in its memory the first feature table, FT1, with musicodes derived from the first measure of the training sequence (Figure 12.11) as follows:

dir = {0, -1, +1, -1}
int = {0, 5, 2, 4}
dur = {120, 120, 120, 120}
ω = 1/1 or 1.0

Then, the system creates FT2 with musicodes extracted from the second measure of the training sequence (Figure 12.12) as follows:

dir = { 0, +1, +1, -1, -1 }
int = { 0, 1, 1, 1, 1 }
dur = { 60, 60, 120, 60, 60 }
ω = 1/2 or 0.5

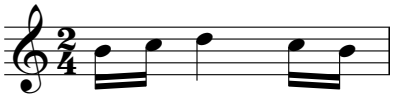

Figure 12.12: The second measure for the example analysis.

Next, MusEng creates FT3, with musicodes from the third measure of the training sequence (Figure 12.13) as follows:

dir = { 0, +1, 0 }
int = { 0, 1, 0 }
dur = { 120, 120, 240 }
ω = 1/3 or 0.33

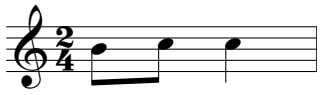

Figure 12.13: The third measure for the example analysis.

The fourth and fifth measures are processes next but MusEng does not create new FTs in these cases because they are repetitions of previous measures; that is, their respective musicodes have already been seen by the system. In this case, only the values of *ω* for the respective FTs are adjusted accordingly. Thus, at this point of the training phase, the *ω* values for each FT are as shown in Table 12.4.

	FT1	**FT2**	**FT3**
ω	1/5 = 0.2	2/5 = 0.4	2/5 = 0.4

Table 12.4: Values of *ω* after three FTs have been created and stored in memory, calculated by dividing the number of instances of a given FT by the total number of FTs analysed.

MusEng's memory after the training phase, complete with 3 FTs is shown in Table 12.5. It is important to stress that particular FTs gain or lose perceptual importance depending on how often the system is exposed to them. Notice, therefore that FT2 and FT3 have higher *ω* values than that of FT1, because they appeared twice.

	dir	***int***	***dur***	***ω***
FT1	0, -1, +1, -1	0, 5, 2, 4	120, 120, 120, 120	0.2
FT2	0, +1, +1, -1, -1	0, 1, 1, 1, 1	60, 60, 120, 60, 60	0.4
FT3	0, +1, 0	0, 1, 0	120, 120, 240	0.4

Table 12.5: MusEng's memory after being trained with the musical sequence shown in Figure 12.5.

12.7.2 Generative Phase

At the generative phase, MusEng generates new FTs by mutating the musicodes of an existing FT towards those of another FT in memory. This process is influenced by the values of *ω*: FTs with larger *ω* values are selected more often than FTs with smaller *ω* values. Note that we wrote: 'tend to be selected'. This is because MusEng uses a Gaussian distribution function to make this selection.

The very first measure of a newly generated structure is typically informed by the first FT in memory (FT1). Let us consider this as the source FT for the mutation. A second FT, the target FT, is selected from memory according to the values held in memory for the variable *ω* – as mentioned above, FTs with

higher ω values tend to be selected as targets more often than FTs with lower ω values.

The generative process is illustrated below by means of a simple example using the memory from the previous learning phase, but considering a mutation on a single musicode only: melodic direction (*dir*). Therefore, let us assume the memory scenario shown in Table 12.6.

	FT1	**FT2**	**FT3**
dir	0, -1, +1, -1	0, +1, +1, -1, -1	0, +1, 0
$\boldsymbol{\omega}$	0.2	0.4	0.4

Table 12.6: A memory scenario with three FTs.

In order to generate a new measure, the *dir* musicode of the source FT1 will be mutated towards the respective musicode values of a target FT. In this case, both FT2 and FT3 have the same ω so there is an equal chance of FT2 or FT3 being selected as the target FT, and a smaller chance of FT1. Let us assume that FT2 is selected as the target. Thus, FT2's *dir* musicode is applied to FT1's *dir* musicode to produce a mutation (represented in bold) as follows:

{0, +1, +1, -1, -1} + {0, -1, +1, -1} = {0, **0**, +1, -1, -1}.

Note that the *dir* musicode has outlying maximum and minimum values of +1 and - 1, hence only the second value is actually mutated (+1) + (-1) = 0. Therefore the newly generated FT contains a *dir* musicode of {0, 0, +1, -1, -1}.

Mutating other musicodes (melody interval, event duration, note pitch, etc.) would yield more variation. Mutations are possible across all musicodes in a similar manner, with the only exception being mutations in modality. These are accomplished by a process of transformation whereby the intervals between successive absolute pitches in the given FTs are forced to conform to pre-set intervals for major, minor, or diminished modes.

Finally, the new FT is rendered into a musical measure (Figure 12.14) and saved into a MIDI file.

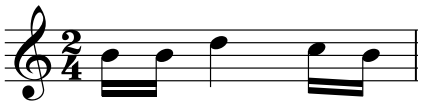

Figure 12.14: The musical rendering of the new FT that was generated by mutating the *dir* musicode from FT1 and FT2.

The above example only showed the generation of a single measure. For longer musical sequences, further FTs are generated by using the next FT in memory as the source FT and mutating it with a target FT that again, is

selected according to the value of the variable *ω* of all other FTs stored in memory.

12.7.3 Transformative Phase

The transformative phase comprises a number of transformation algorithms that modify a given musical sequence, three of which will be explained in this section.

Although there are some differences in the specific processing undertaken by each algorithm, the basic signal flow is quite similar for all of them. The generated input signal is modified towards values given by one of the transformation algorithms. With most of the transformation algorithms, the amount of modification is scaled according to the fMRI data. The fMRI data, or more specifically the data extrapolated from the fMRI scans by ICA analysis, is referred to as the *fMRI_index*. This data is provided to MusEng on a ten-point scale with values between 0 and 9. In order to use the fMRI index as a control signal (*CS*) for the transformation algorithms, MusEng first scales the data to a range between 0.1 and 1.0. The system applies the following simple scaling process to the value of the *fMRI_index* (Equation 12.2).

$$CS = \{(fMRI\_index + 1) * 0.1\} \quad (12.2)$$

A difference value *d* between the input and the transformed musicodes is also calculated. This difference is then multiplied by the *CS* to give a final scaled modifier value: *SMV*. The *SMV* is summed with the input signal to directly transform the output. This gives a degree of fMRI-controlled variability in each transformation: a high *fMRI_index* value will result in larger transformations to the music, whereas a low *fMRI_index* value will result in smaller transformations.

Below are examples of three of the transformation algorithms, which illustrate the effect of varying the *fMRI_index*: pitch inversion, pitch scrambling, and pitch delta.

12.7.3.1 Pitch inversion algorithm

Given an input musical sequence, the pitch inversion algorithm creates a new sequence, which is effectively the input sequence turned upside-down. For instance, a sequence rising in pitch would descend in pitch after being passed through this transformation. In order to illustrate this, let us consider the measure produced in generation phase example, as shown in Figure 12.14. Incidentally, this measure will be used as the starting point for the following two transformation examples as well.

The melody interval musicode for this measure is {0, 0, 3, 2, 1} and the note pitch musicode is {B4, B4, D5, C5, B4}. In this case the MIDI values are 71, 71, 74, 72 and 71, respectively; MIDI uses a range of 128 pitch values. There are a variety of ways to accomplish a pitch inversion, including diatonic and chromatic options, or inversions around a specific sounding pitch. MusEng processes pitch inversion simply by subtracting the current MIDI pitch value from 128, and substituting in the resulting natural number as the new pitch value. For instance, the transformed pitch values for our example created using this technique would be as follows: (128 - 71 = 57), (128 - 71 = 57), (128 - 74 = 54), (128 - 72 = 56) and (128 - 71 = 57).

The resulting MIDI values are 57, 57, 54, 56 and 57, yielding the following pitch sequence {A3, A3, F#3, G#3, A3}. Note that the inverted sequence maintains the original melody interval musicode of {0, 0, 3, 2, 1}, whilst giving an upside down melody, as shown in Figure 12.15.

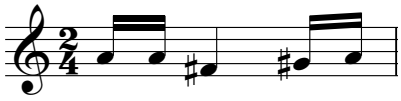

Figure 12.15: Newly inverted sequence, after transformation of measure in Figure 12.14.

The example above assumed a maximal fMRI index value of 9, which once scaled to create a *CS* gives 1.0. However, as mentioned in the introduction to this section, varied degrees of transformations is also possible by scaling the amount of transformation according to the value of the *fMRI_index*. The difference between the input and the transformed pitches is multiplied by the *CS*, before being summed with the input to create the final transformed output value (Equation 12.3).

$$New\_pitch = \{Input\_pitch + ((Input\_pitch - transf\_pitch) * [(fMRI\_index + 1) * 0.1])\} \quad (12.3)$$

Let us examine what happens if we assume an *fMRI_index* equal to 5, which yields a *CS* equal to 0.6. In this case, we would expect an output approximately half way between the original pitch and the inversion; in other words, an almost neutral set of intervals. First, the difference *d* between the maximal inversion and the input signal for each of the musicode values needs to be calculated as follows:

$d = \{(57 - 71), (57 - 71), (54 - 74), (56 - 72), (57 - 71)\}$
$d = \{-14, -14, -20, -16, -14\}$

Then, the scaled modifier values are calculated by multiplying the difference values by the value of *CS*:

$SMV = \{(-14 * 0.6), (-14 * 0.6), (-20 * 0.6), (-16 * 0.6), (-14 * 0.6)\}$

SMV = {-8.4, -8.4, -12, -9.6, -8.4}

Finally, the *SMV* values are summed with the original input to give a transformed set of output values:

New_pitches = {(71 - 8.4), (71 - 8.4), (74 – 12), (72 - 9.6), (71 - 8.4)}
New_pitches = {62.6, 62.6, 62, 62, 62.6}

Pitch values are rounded up to the nearest whole number as per the MIDI standard, giving a transformed set of pitch values equal to {63, 63, 62, 62, 63}, which is rendered as {D#4, D#4, D4, D4, D#4}, as shown in Figure 12.16.

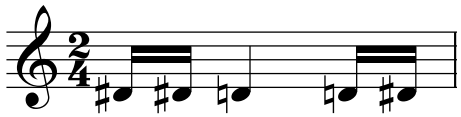

Figure 12.16: Sequence after inversion with *fMRI_index* = 5, giving a nearly neutral set of pitch intervals.

12.7.3.2 Pitch scrambling algorithm

In simple terms, the pitch scrambling algorithm orders the pitch values of the input signal into a numerical list, which is then re-ordered randomly. This provides a stochastic component to the transformation algorithm. Using the same measure as for the previous example (Figure 12.14) as a starting point, let us examine the result of applying this transformation. The process is as follows:

- Input pitches: {71, 71, 74, 72, 71}
- Order pitches in ascending order: {71, 71, 71, 72, 74}
- Scramble the order of pitches randomly: {74, 72, 71, 71, 71}
- Output pitches: {74, 72, 71, 71, 71}

In this case, the output would be rendered as {D5, C5, B4, B4, B4}. Re-running the transformation, a further three times would give further variants, for example: {72, 74, 71, 71, 71}, {71, 74, 72, 71, 71} and {71, 74, 71, 72, 71}, rendered as {C5, D5, B4, B4, B4}, {B4, D5, C5, B4, B4} and {B4, D5, B4, C5, B4}, respectively, as illustrated in Figure 12.17.

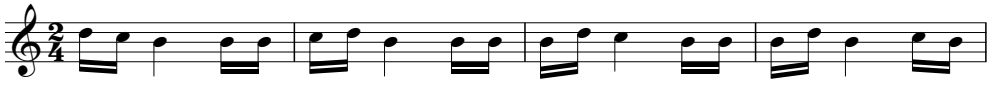

Figure 12.17: The result from applying the pitch scrambling algorithm four times on the same input.

As with the pitch inversion algorithm, the value of *fMRI_index* can be used to create a control signal with which the amount of transformation can be varied.

In order to illustrate this, let us assume an *fMRI_index* equal to 3. This gives a *CS* value of 0.4.

Considering the same input measure as before (Figure 12.14) and the transformed values from the first pitch scramble shown in Figure 12.17, the value of *d*, between the first scramble and the input signal is calculated as follows:

d = {(74 – 71), (72 – 71), (71 – 74), (71 – 72), (71-71)}
d = {3, 1, -3, -1, 0}

The scaled modifier values are then calculated by multiplying the difference values by *CS* = 0.4:

SMV = {(3 * 0.4), (1 * 0.4), (-3 * 0.4), (-1 * 0.4), (0 * 0.4)}
SMV = {1.2, 0.4, -1.2, -0.4, 0}

Finally, the *SMV* values are summed with the values of the original input to give a transformed set of output values:

New_pitches = {(71+1.2), (71+0.4), (74-1.2), (72-0.4), (71-0)}
New_pitches = {72.2, 71.4, 72.8, 71.6, 71}

As before, pitch values are rounded up to the nearest whole number as per the MIDI standard, giving a transformed set of pitch values equal to {72, 71, 73, 72, 71}, which is rendered as {C5, B4, C#5, C5, B4}, as shown in Figure 12.18. Note that the output is significantly closer in overall structure to the unscrambled input than the first scrambled transformation shown in Figure 12.17, with only the first and third notes having changed here.

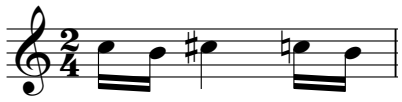

Figure 12.18: Transformed output created by pitch scrambling algorithm assuming *fMRI_index* = 3.

12.7.3.3 Pitch delta algorithm

Pitch delta represents the rate of change in the pitch values in each measure. The algorithm works by calculating the difference between the initial pitch and the successive pitch in each pair of notes. The pitch delta value is then used to transform the input pitch by summing the two values together (Equation 12.4).

$$New\_pitch = \{(successive\_pitch - initial\_pitch) + initial\_pitch\} \quad (12.4)$$

Assuming the same input as for the previous examples (Figure 12.14), with a string of note pitch values equal to {71, 71, 74, 72, 71}, the delta values for this transformation are calculated as follows:

1) *Delta_1* = {(71 – 71) + 71}
Delta_1 = 0
New_pitch_1 = 71 (no change)

2) *Delta_2* = {(71 – 71) + 71}
Delta_2 = 0
New_pitch_2 = 71 (no change)

3) *Delta_3* = {(74 – 71) + 74}
Delta_3 = 3
New_pitch_3 = 77

4) *Delta_4* = {(74 – 72) + 74}
Delta_4 = 2
New_pitch_4 = 76

5) *Delta_5* = {(72 – 71) + 72}
Delta_5 = 1
New_pitch_4 = 73

The transformed output would therefore comprise a pitch string of {71, 71, 77, 76, 73}, which is rendered as {B4, B4, F5, E5, C#5}, as shown in Figure 12.19. Thus, the application of the pitch delta algorithm gives the effect of exaggerating the melodic intervals from a given measure; large intervals become even larger, whilst melodies with little or no interval between successive notes remain unchanged.

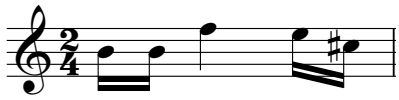

Figure 12.19: Transformed output created by the pitch delta algorithm.

As with the previous cases of transformations, the above example assumed a maximal *fMRI_index* value, but the effect of the transformation can be mediated by reducing the value of the *fMRI_index*. This is illustrated in the following example.

Let us assume the case of *fMRI_index* = 2. This gives a *CS* value of 0.3. With such a low value for the control signal, we should expect only a small amount of pitch delta transformation.

As before, we will use the input signal shown in Figure 12.14, this time with the transformed values from the full pitch delta transformation shown in Figure 12.19, to the difference *d*, as follows:

d = {(71 – 71), (71 – 71), (77 – 74), (76 – 72), (73 – 71)}
d = {0, 0, 3, 4, 2}

The scaled modifier values are then calculated by multiplying the difference values by the *CS* value, which is equal to 0.3:

SMV = {(0 * 0.3), (0 * 0.3), (3 * 0.3), (4 * 0.3), (2 * 0.3)}
SMV = {0, 0, 0.9, 1.2, 0.6}

Finally, the *SMV* values are summed to the original input to give a new sequence of pitch values:

New_pitches = {(71 + 0), (71 + 0), (74 + 0.9), (72 + 1.2), (71 +0.6)}
New_pitches = {71, 71, 74.9, 73.2, 71.6}

As with the previous transformation examples, pitch values are rounded up to the nearest whole number, giving a transformed sequence of pitch values of {71, 71, 75, 73, 72}, which is rendered as {B4, B4, D#5, C#5, C5}, as shown in Figure 12.20. The exaggerating effect of the pitch delta has been radically mediated by the value of *CS*, with a much smaller amount of change seen in the transformed output than in the full pitch delta transformation shown in Figure 12.19.

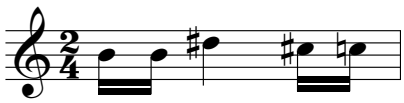

Figure 12.20: Transformed output created by pitch delta algorithm with a relatively low *fMRI_index* value of 3.

The generative potential of a composition system that incorporates transformative processes, such as that offered by MusEng, is high.

12.8 Concluding Remarks

Research into BCMI often is devoted to technical aspects of building BCMI systems, which is not surprising giving the plethora of technical difficulties that need to be addressed to implement a decent system. In this chapter, however, we ventured to explore the creative potential of the science and technology behind BCMI research: Music Neurotechnology.

We introduced an approach to music composition informed by the notion that the neural patterns and the corresponding mental images of objects and events around us are creations of the brain prompted by the information we receive through our senses. In the case of music, even though humans have

identical mechanisms for processing the basics of sound, music as such is a construction of the brain. Indeed, there is increasing hard evidence that this construction differs from person to person. When we listen to music, sounds are deconstructed as soon as they enter the ear. Different streams of neuronally coded data travel through distinct auditory pathways towards cortical structures, such as the auditory cortex and beyond, where the data are reconstructed and mingled with data from other senses and memories, into what is perceived as music.

Metaphorically speaking, the compositional approach that we developed to composer *Symphony of Minds Listening* did to the Beethoven score what our hearing system does when we listen to music: sounds are deconstructed as they enter the ear and are relayed through various pathways towards cortical structures, where the data is then reconstructed into what is perceived as music.

We would like to highlight that the composition of the piece evolved in tandem with the development of the MusEng software. Some of MusEng's functionalities were firstly applied manually to compose a section of the piece, before they were implemented in software to aid the composition of other sections. The compositional practice therefore informed the design of the software, and the design of the software influenced the compositional practice. We believe that this is an important shift of paradigm from most scenarios of using computers in music. A piece of software is often developed from abstract specifications and tested only after it has been almost fully implemented. Moreover, composers are often confronted with software that does not always do what they need to do. Our paradigm to systems development may not as cost-effective as more standard methods, as it requires much more time to develop. However, it opens a significant opportunity for composers to actively participate in the design process. As we continue developing this work, more and more procedures emerging from the left hand side of the block diagram in Figure 12.5 will certainly make its way to the right hand side.

We believe that Music Neurotechnology provides musicians with an unprecedented opportunity today to develop new approaches to music that would have been unthinkable a few years ago. This chapter unveiled only the tip of the iceberg.

12.9 Questions

1. How would you define the emerging field of Music Neurotechnology?

2. What do you understand by the 'mind as music' hypothesis?

3. Explain the metaphor that *Symphony of Minds Listening* is intended to express artistically.

4. What is the point of composing each movement of the symphony based on the fMRI scan of a different person?

5. Explain what it ICA and its role in the project presented in this chapter.

6. What are the approaches to using computer-generated materials in musical composition discussed in this chapter? Discuss the differences between them, and the advantages and disadvantages of each approach.

7. Can you envisage an approach to use computers in music beyond the ones discussed in this chapter?

8. What is the rational for dividing Beethoven's piece into 13 sections before processing it with MusEng?

9. Why MusEng apply transformations to the music?

10. Create a new kind of transformation for MusEng and explain it in detail.

12.10 References

Ashburner, J. and The FIL Methods Group at UCL (2013). *SMP8 Manual*. London: Insitute of Neurology, University College London. Available online at http://www.fil.ion.ucl.ac.uk/spm/ (Assessed on 04 November 2013).

Bremment, J. D. (2005). *Brain Imaging Handbook*. London: W. W. Norton & Co.

Churchland, P. (2007). *Neurophilosophy at Work*. Cambridge: Cambridge University Press.

Eysenck, M. W. and Keane, M. T. (2005). *Cognitive Psychology: A Student's Handbook*. Hove and New York: Psychology Press (Taylor and Francis).

Lloyd, D. (2011). "Mind as music", *Frontiers in Psychology* 2:63. DOI: 10.3389/fpsyg.2011.00063.

MathWorks (2014). *Matlab: The Language of Technical Computing*. Website: http://www.mathworks.co.uk/ (Assessed on 04 November 2013).

Miranda, E. R. (2014). *Thinking Music*. Plymouth: University of Plymouth Press.

Miranda, E. R. (2013). “On Computer-aided Composition, Musical Creativity and Brain Asymmetry”. In D. Collins (Ed.), *The Act of Musical Composition*. Farnham: Ashgate.

Miranda, E. R. and Gimenes, M. (2011). “An Ontomemetic Approach to Musical Intelligence”. In E. R. Miranda (Ed.), *A-Life for Music: Music and Computer Models of Living Systems*. Middleton, WI: A-R Editions. Pp. 261-286.

Miranda, E. R. (Ed.) (2000). *Readings in Music and Artificial Intelligence*. Abingdon: Routledge.

Stone, J. V. (2004). *Independent Component Analysis: A Tutorial Introduction*. Cambridge, MA: The MIT Press.